RT5 - Zephyr Real Time Programming

Real-time programming applied to the Zephyr operating system

Objectives

- Learn how to develop, configure, debug and trace Zephyr applications
- Discover the real time multitasking concept
- Understand Real Time constraints, like determinism, preemption or interrupts
- Understand the Zephyr kernel Services
- Learn communication and synchronization mechanisms
- Interactions with processor architecture features
- Understand Zephyr memory management and data structures
- Understand User mode and kernel mode
- Writing a device tree
- Writing a complete driver

Course Environment

- Theoretical course
  - PDF course material (in English) supplemented by a printed version.
  - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance.
- Practical activities
  - Practical activities represent from 40% to 50% of course duration.
  - Code examples, exercises and solutions
  - One PC (Linux ou Windows) for the practical activities with, if appropriate, a target board.
    - One PC for two trainees when there are more than 6 trainees.
  - For onsite trainings:
    - An installation and test manual is provided to allow preinstallation of the needed software.
    - The trainer come with target boards if needed during the practical activities (and bring them back at the end of the course).
- Downloadable preconfigured virtual machine for post-course practical activities
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

Prerequisites

- Good C programming skills (see our L2 - C language for Embedded MCUscourse)

Duration

- Total: 5 days
- From 40% to 50% of training time is devoted to practical activities
- Some Labs may be completed between sessions and are checked by the trainer on the next session

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Evaluation modalities

- The prerequisites indicated above are assessed before the training by the technical supervision of the traineein his company, or by the trainee himself in the exceptional case of an individual trainee.
- Trainee progress is assessed in two different ways, depending on the course:
  - For courses lending themselves to practical exercises, the results of the exercises are checked by the trainer while, if necessary, helping trainees to carry them out by providing additional details.
  - Quizzes are offered at the end of sections that do not include practical exercises to verifythat the trainees have assimilated the points presented
- At the end of the training, each trainee receives a certificate attesting that they have successfully completed the course.
  - In the event of a problem, discovered during the course, due to a lack of prerequisites by the trainee a different or additional training is offered to them, generally to reinforce their prerequisites,in agreement with their company manager if applicable.

Plan

First Day

Real-Time Concepts

- Base real time concepts
- The Real Time constraints
- Multi-task and real time
- Tasks and Task Descriptors
  - Content of the task descriptor
  - List of task descriptors
- Context Switch

Introduction to Zephyr

- Zephyr Project
- Zephyr Ecosystem
- Why use Zephyr
- Install and use Zephyr
- Build and Configuration Systems
  - West
  - CMake
  - Kconfig and configuration overlay
  - Configuration tools: menuconfig and guiconfig

Zephyr Without Threads

- Operation without Threads
- GPIO
- Random Number Generation
- Utilities
- Data Structures
  - Single-linked List
  - Double-linked List
  - Ring Buffers

Exercise: *Hello World from Zephyr, configure and blink LEDs using Zephyr*
Exercise: *Manage Zephyr linked list and understand container_of macro*

Scheduling

- Task Scheduling and Preemption
  - Tick based or tickless scheduling

- Scheduling systems and schedulability proof
  - Fixed priorities scheduling
  - RMA and EDF scheduling
- Scheduling through Zephyr
  - Scheduling Algorithm
  - Cooperative Time Slicing
  - Preemptive Time Slicing

Second Day

Thread Management

- Thread Control Block
- Creating Threads
- Threads Priorities
- Thread States
- Main and Idle Threads
- Delays
- Changing Thread Priority
- Suspending Threads
- Kernel Structures
  - Simple linked-list ready queue
  - Red/black tree ready queue
  - Traditional multi-queue ready queue
- Thread Custom Data
- Scheduling Traces
  - Runtime Statistics
  - User-Defined Tracing
  - Percepio Tracealyzer

Exercise: *Create and manage threads*
Exercise: *Create periodic threads*
Exercise: *Create config overlay for visual trace diagnostics using Tracealyzer*

Memory Management in Zephyr

- Memory Managers
- Dynamic memory managers
  - K_heap
  - System heap
  - Memory Slabs
  - Memory Blocks
- Heap Listeners
- Thread Resource Pools
- Stack Overflow detection
- User Mode
- Memory Domains

Exercise: *Understand dynamic memory allocation in Zephyr*
Exercise: *Display threads information and detect stack overflow*

Third Day

Resource Management

- Mutual Exclusion
- Critical Sections
- Mutexes
- Gatekeeper threads
- Atomic

- Lock-Free Data Structures
- SpinLocks

Exercise: *Implement mutual exclusion between threads*

Synchronization Primitives

- Synchronization
- Semaphores
- The Readers/Writer Problem
- Condition variables
- Events and Event Groups
- Polling

Exercise: *The producer-consumer problem, synchronize and avoid concurrent access problems*
Exercise: *Understanding event bit group by synchronizing several threads*

Fourth Day

Data Passing

- Message Queues
- Pipes
- Queues
  - FIFOs
  - LIFOs
- Mailboxes
- Stacks
- Zephyr Bus (Zbus)

Exercise: *Create a print gatekeeper thread using message queue*

Interrupt Management

- Threads and Interrupts
- Interrupts in zephyr
- Interrupts on ARM Cortex-M
- Handler thread
- Queue within an ISR
- Workqueue Threads
- Power Management

Exercise: *Understand how to wait on multiple events and interrupt safe APIs*
Exercise: *Understand how to pass data using Queues from an interrupt to a thread*
Exercise: *Create and submit work items from interrupts to custom WorkQueue*

Software Timers

- Timers
  - Defining a Timer
  - Using a Timer Expiry Function
- Timer types
  - One-shot timers
  - Auto-reload timers
- Timer Commands

Exercise: *Understand the use of one-shot and auto-reload timers*

Fifth Day

Modules

- Why to use modules?

- Module structure
- Out-of-tree module
- YAML files
- Module CMakeLists.txt
- How to add and use custom Kconfigs

Exercise: *Create a simple hello world module*
Exercise: *Create a module that uses custom Kconfig options*

Zephyr device driver model

- Introduction to Device Drivers
- Overview of the Zephyr device driver model
- Standard Drivers
- The struct device
- Subsystems
- API Extensions
- Initialization Levels
- Dependencies between device drivers
- Define devices programmatically
- Adding In-Tree Code to Zephyr Source Code

Exercise: *Create a driver that respects the Zephyr Device Driver Model and define devices*
Exercise: *Writing in-tree drivers*

Device Trees in Zephyr

- Overview of Device Tree (DT) and its role in Zephyr
- Device Tree VS Kconfig
- Device Tree node structure
- Device Tree bindings
- Overlay and yaml files
- APIs to access device tree properties
- Write device drivers using device tree APIs
- Device Tree in Zephyr VS Linux

Exercise: *Create a driver that uses custom device tree and Kconfig*

Renseignements pratiques

Duration : 5 days
Cost : 3070 € HT